# Peer-to-peer solution of 2D cutting stocks problems

Didier El Baz, Mhand Hifi, Toufik Saadi

# Peer-to-peer solution of 2D cutting stocks problems

Didier Elbaz, Mhand Hifi, and Toufik Saadi

CNRS, LAAS, 7, avenue du Colonel Roche
Univ. de Toulouse, LAAS, F-31400 Toulouse, France
EPROAD Lab.
Univ. de Picardie,33, rue Saint Leu F-80039 Amiens Cedex 1, France
elbaz@laas.fr,{hifi,toufik.saadi}@u-picardie.fr

**Abstract.** Peer-to-peer (P2P) applications have known great developments these years. These applications were originally designed for file sharing and are now considered to a larger scope from video streaming to system update and distributed data base. Recent advances in microprocessors architecture and networks permit one to consider new applications like High Performance Computing (HPC). In this paper, we study the parallel solution of 2D cutting stock problems with the peer-to-peer P2PDC environment. First, we present briebly the decentralized version of P2PDC. Then, we propose a distributed P2P algorithm based on dynamic programming and beam search. Finally, we display and analyze computational results.

**Keywords:** parallel computing, distributed computing, peer-to-peer computing, 2D cutting stock problems, beam search, algorithm

## 1 Introduction

Peer-to-peer (P2P) applications have known great developments these years. Recent advances in microprocessors architecture and networks permit one to consider new applications like High Performance Computing (HPC). Therefore, we can identify a real stake at developing new protocols and environments for HPC since this can lead to economic and attractive solutions. Nevertheless, task parallel model and distributed iterative methods for large scale optimization on P2P networks gives raise to numerous challenges like communication management, scalability and peer volatility; moreover, one has to cope with heterogeneity of machines. In [1], a peer-to-peer Self Adaptive communication Protocol (P2PSAP) which is suited to high performance distributed computing has been proposed. P2PSAP chooses dynamically the most appropriate communication mode between any peers according to decisions taken at application level like scheme of computation and elements of context like topology at transport level. In [2], a centralized version of the P2PDC, environment for high performance peer-to-peer computing which makes use of P2PSAP protocol in order to allow

direct communication between peers has been presented. P2PDC is devoted to task parallel applications. In [2], a first series of computational results obtained for a numerical simulation problem on the NICTA testbed has been displayed and analyzed. In this paper, we study the distributed solution of 2D cutting stock problems with the decentralized version of P2PDC.

This paper is structured as follows. Section 2 deals with related work. Section 3 deals with the 2D cutting stock problems. In section 4, we introduce some methods for solving 2D cutting stock problems and we present the parallel algorithm. Computational results are displayed in Section 5.

## 2 Related work

Recently, middleware like BOINC [3] or OurGrid [4] have been developed in order to exploit the CPU cycles of computers connected to the network. Those systems are generally dedicated to applications where tasks are independent and direct communication between machines is not needed. MapReduce [**?**] is a programming model and an associated implementation for processing and generating large data sets. This programming model is not appropriate for distributed iterative algorithms with frequent communication between peers.

P2P-MPI [5] is a framework aimed at the development of message-passing programs in large scale distributed networks of computers. P2P-MPI is developed in Java and makes use of Java TCP socket to implement the MPJ (Message Passing for Java) communication library. P2P-MPI implements a fault tolerance approach using peer replication that may be not efficient and not appropriate to P2P context and connected problems because the number of peers involved in the computation will multiply; furthermore, the coordination protocol insuring coherence between replicas has great overhead.

## 3 2D cutting stock problems

Several industrial applications require cutting or packing the largest number of items into a rectangular unit so as to minimize the waste of the rectangular stock unit or to maximize the packed pieces. These cutting (or packing) problems are difficult combinatorial optimization problems, known in the literature as 2D cutting stock problems [7] [9]. The problem consists of cutting, from a large rectangle $R$ of dimensions $L \times W$, a number of very small rectangle types (or pieces or items) $i$, $i \in I = \{1, \ldots, n\}$, where item $i$ is characterized by its dimensions $l_i \times w_i$, its demand $b_i$, and results in a profit $c_i$. Item $i$, $i \in I$ has a fixed orientation; that is, an item of dimensions $l \times w$ is different from an item of size $w \times l$ when $l \neq w$. In addition, in the final cutting pattern, each piece is produced by at most two guillotine cuts. A guillotine cut divides a rectangle from one of its edges to the opposite edge while being parallel to the two remaining edges, so each item in the final cutting patternis is obtained by two guillotine cuts:

(i) $R$ is first divided into a set of *horizontal strips*, and
(ii) each generated horizontal strip is subsequently considered individually and chopped across its width.

## 4 Solving 2D cutting stock problems

### 4.1 Strip generation

This method generates a set of *general optimal horizontal* strips. For a given strip of dimensions $(L, \omega)$, where $\omega \in \{w_1, \ldots, w_n\}$, this method generates a general optimal horizontal strip −with pieces whose widths are less than or equal to the strip's width− according to the optimal solution of the following bounded knapsack problem:

$$BK_{L,\omega} \begin{cases} f_\omega(L) = \max \displaystyle\sum_{i \in S_\omega} c_i x_i \\ \text{subject to} \quad \displaystyle\sum_{i \in S_\omega} l_i x_i \leq L \\ \qquad\qquad x_i \leq b_i,\ x_i \text{ integer},\ i \in S_\omega, \end{cases}$$

where $S_\omega = \{k \in I \mid w_k \leq \omega\}$ denotes the set of pieces assigned to the strip $(L, \omega)$, $x_i$ denotes the number of times piece of type $i$ appears in $(L, \omega)$ without exceeding the upper demand $b_i$ of piece type $i$. Finally, $f_\omega(L)$ is the solution value of strip $(L, \omega)$.

Let $\overline{w}_1 < \ldots < \overline{w}_n$ be the set of distinct widths of the $n$ pieces; that is, for $i = 1, \ldots, n,\ w_i \in \{\overline{w}_1, \ldots, \overline{w}_n\}$. As detailed in [6], solving $BK_{L,\overline{w}_n}$, using dynamic programming, generates all general optimal strips of width $\omega = 1, \ldots, \overline{w}_n$.

### 4.2 Beam search for 2D cutting stock problems

Beam search is a truncated tree search procedure that was introduced in the context of scheduling [10], and has since been successfully applied to many other combinatorial optimization problems. It avoids exhaustive enumeration by performing a partial search of the solution space. In fact, at each level of the search tree, a subset of elite nodes is selected for further branching whereas the complementary subset of nodes is discarded forever. The selected *set of elite nodes* has at most $\beta$ nodes where $\beta$ is a prefixed *beam width*.

As with branch and bound, a lower bound can be used to fathom nodes. Indeed, if an initial feasible solution is available, then it is set as the incumbent solution and its value is assigned to $z^\star$. On the other hand, if no initial feasible solution is available, $z^\star$ is set to $-\infty$. Each node of $B$ (which is the set of nodes to be further investigated) generates a set of offspring nodes, and appends them to $B_\beta$. If a node $\mu$ of $B_\beta$ is a leaf (i.e, no further branching is possible out of $\mu$), then its objective function value $z_\mu$ is computed and compared to $z^\star$. If $z_\mu > z^\star$, then the incumbent solution is set to the leaf node; $z^\star$ is then updated: $z^\star = z_\mu$; and $\mu$ is removed from $B_\beta$. The nodes of $B_\beta$ are assessed using an evaluation

operator, and ranked in a non-ascending order of their values. The first $\beta$ nodes of $B_\beta$ are then chosen as the elite nodes and transferred to $B$; whereas the remaining nodes of $B_\beta$ are fathomed resulting in $B_\beta$ being reset to the empty set. This process is reiterated until no further branching is possible; that is, until $B = \emptyset$. For more details, we invite the reader to consult [6] and [11].

### 4.3 Parallel algorithm

The parallel algorithm explores in parallel $\eta$ nodes of the developed tree. Each peer guides its search-resolution process by using a list of nodes that have not yet been explored, and uses a best-first search strategy for selecting the successive sets of elite nodes. In the selection phase, nodes are sorted in a specific order. When a new peer arrives and declares itself in the P2P framework, Each *machine* $k,\ k = 1, \ldots, \eta$, sends to it, its internal lists. The *new peer* enters the nodes in its *fist internal list* and selects a starting node $\mu_k = \big[\big((L, W - Y),\ (L, Y)\big);\ \mathbf{b}^{res}_{\mu_k}\big]$, whose partial feasible solution's value is $z^{Local}_{\mu_k}$. The new peer refines the complementary upper bound by solving $BK_{L,Y}$ with the demand for piece type $i$ set to its counterpart in $\mathbf{b}^{res}_{\mu_k}$, for all $i \in I$ such that $w_i \leq \overline{w}_r \leq Y < \overline{w}_{r+1}$.

When branching out of $\mu_k$, the new peer creates $r$ nodes. For each node $\nu_j,\ j = 1, \ldots, r$, the new peer $k$ packs the general strip $(L, \overline{w}_j)$, which is the optimal solution to $BK_{L,\overline{w}_j}$, into the current sub-rectangle $(L, Y)$, and computes the residual demand $\mathbf{b}^{res}_{\nu_j}$. It sets $\nu_j = \big[\big((L, W - Y + \overline{w}_j), (L, Y - \overline{w}_j)\big);\ \mathbf{b}^{res}_{\nu_j}\big]$, assesses $z^{Local}_{\nu_j}$, and evaluates $\overline{U}_{\nu_j}$. Next, the new peer $k$ stores the $r$ offspring nodes of $\mu_k$ into a secondary internal list $\overline{B}_k$, and save the best $min\{\beta, |\overline{B}_k|\}$ elite nodes of $\overline{B}_k$ to $B_k$, where the elite nodes are chosen.

## 5 Computational results

This section evaluates the effectiveness of the proposed distributed algorithm (noted P2PGBS) by testing it on set of instances extracted from [6]. The dataset contains six large instances. The algorithm is run on Grid5000 platform with P2PDC framework.

The optima of these instances are unknown; thus, for each instance, the solution obtained by P2PGBS is compared to the best lower bound provided by the following algorithms: Cplex solver (v.9), the algorithm PAR of [12], and SBS of [6] with $\beta = 2$ and 4. The time limit is fixed to 3000 seconds for the Cplex solver and PAR (as in [8] and [6]), and to 900 seconds for SBS.

The analysis indicates that P2PGBS yields better results than Cplex, PAR and SBS. It improves six best solutions of the literature. These results show that another factor impact P2PGBS. Our investigations shows that frequency of processor's connection/disconnection impacts P2PCGBS.

## 6 Acknowledgments

Part of this study was made possible with the support of ANR−07−CIS7−11 funding and Grid5000

## References

1. Didier El Baz and The Tung Nguyen, A self-adaptive communication protocol with application to high performance peer-to-peer distributed computing, Proceedings of the 18th Euromicro conference on Parallel, Distributed and Networked-Based Processing,IEEE CPS, p. 323-333. 2010
2. The Tung Nguyen and Didier El Baz and Pierre Spiteri and Guillaume Jourjon and Minh Chau, High Performance Peer-to-Peer Distributed Computing with Application to Obstacle Problem, Proceedings of HOTP2P/IPDPS2010, 2010
3. David P. Anderson, Boinc: A system for public-resource computing and storage, 5th IEEE/ACM International Workshop on Grid Computing, 2004
4. Nazareno Andrade and Walfredo Cirne and Francisco Brasileiro and Paulo Roisenberg, OurGrid: An approach to easily assemble grids with equitable resource sharing, Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing, 2003
5. Stephane Genaud and Choopan Rattanapoka, A Peer-to-Peer Framework for Message Passing Parallel Programs, IOS Press, Fatos Xhafa, Vol.17, p. 118–147,2009
6. M. Hifi, R. M'Hallah, T. Saadi, Algorithms for the constrained two-staged two-dimensional cutting problem, *INFORMS Journal on Computing, Vol.20, p.212–221, 2008.*
7. *H. Dyckhoff, A typology of cutting and packing problems,* European Journal of Operational Research, Vol.44, p.145–159, 1990.
8. R. Alvarez-Valdes, R. Martì, A. Parajón, J.M. Tamarit, GRASP and path relinking for the two-dimensional two-staged cutting stock problem, *INFORMS Journal on Computing, Vol.19, p.1–12, 2007.*
9. *G. Wäscher, G., Haussner, H. Schumann, An improved typology of cutting and packing problems,* European Journal of Operational Research, Vol.183, p.1106-1108, 2007.
10. P.S. Ow, T.E. Morton, Filtered beam search in scheduling, *International Journal of Production Research, Vol.26, p.297–307, 1988.*
11. *M.Hifi and T.Saadi : A parallel algorithm for constrained two-staged two-dimensional cutting problems. Computational Optimization and Applications, DOI 10.1007/s10589-010-9351-5, issn 0926-6003, 2010.*
12. *M. Hifi, R. M'Hallah, Strip generation algorithms for two-staged two-dimensional cutting stock problems,* European Journal of Operational Research, Vol.172, p.515–527, 2006.